P2 mesh optimization operators

Rémi Feuillet, Adrien Loseille, Frédéric Alauzet

P^2 mesh optimization operators

Rémi Feuillet, Adrien Loseille and Frédéric Alauzet

Abstract Curved mesh generation starting from a P^1 mesh relies on mesh deformation and mesh optimization techniques. Mesh optimization techniques consist in locally modifying the mesh in order to improve it with respect to a given quality criterion. This work presents the generalization of two mesh quality-based optimization operators to P^2 meshes. The generalized operators consist in mesh smoothing and generalized swapping. With the use of these operators, P^2 mesh generation starting from a P^1 mesh is more robust and P^2 connectivity-change moving mesh methods for large displacements are now possible.

1 Introduction

In numerical simulation, unstructured meshes are commonly used. More specifically, in Computational Fluid Dynamics (CFD) they are used to help to solve real world problems found in industry and government. In the last decade high-order resolution methods (continuous Galerkin [4], discontinuous Galerkin [10], spectral differences [19], ...) have been used. To preserve the high-order of convergence of these methods, it is required to have a high-order representation of the geometry in the mesh. These meshes are curved in order to fit at best with the boundary of the studied geometric shape. In this context, the generation and the processing of high-order meshes is necessary. To generate high-order meshes, several approaches exist. Some are using a PDE or variational approach to curve a P^1 mesh into a P^k mesh [5, 9, 14], others are based on optimization and smoothing operations and start from a P^1 mesh with a constrained P^k curved boundary in order to generate a suitable P^k mesh [12, 15, 16]. In all these techniques, the key feature is to find the best deformation to apply to the P^1 mesh and to optimize it.
A connectivity-change moving-mesh method [1] that enables closed-advancing boundary layer mesh generation [2] relies on both mesh deformation and mesh optimization techniques. In [1], the motion of vertices is first computed thanks to a linear elasticity model and then the position of these vertices is changed via local mesh optimization operators such as generalized swap-

Rémi Feuillet
GAMMA3 team, INRIA Saclay, France
POEMS team, CNRS/ENSTA/INRIA, France, e-mail: remi.feuillet@inria.fr

Adrien Loseille
GAMMA3 team, INRIA Saclay, France, e-mail: adrien.loseille@inria.fr

Frédéric Alauzet
GAMMA3 team, INRIA Saclay, France, e-mail: frederic.alauzet@inria.fr

ping and mesh smoothing. To apply this connectivity-change moving-mesh method to high-order meshes, these two operators need to be generalized to high-order meshes.
In this paper, P^2 mesh quality-based optimization operators are presented. They are a generalization of the P^1 operators and rely on the resolution of a local optimization problem. These two operators can be applied to improve P^2 mesh generation starting from a P^1 and enable high-order connectivity-change moving mesh methods.
The paper is outlined as follow. Section 2 defines what a high-order mesh and sets up validity and quality criteria. Section 3 deals with P^2 mesh optimization. Section 4 shows two applications with examples of P^2 mesh optimization. The first one is an improvement of a P^2 mesh generation technique that curves a P^1 mesh thanks to a high-order linear elasticity solver and the second one describes a P^2 connectivity-change moving mesh method also using a high-order linear elasticity solver. Finally, Section 5 deals with conclusions and perspectives driven by this work.

2 High-order element, validity and quality criteria

To properly define P^2 quality-based optimization operators, it is fundamental to properly define quality and validity criteria for high-order elements.
In general, a finite element is defined [3] by the triplet $\{K, \Sigma_K, V_h\}$ where K denotes the geometric element (triangle, etc), Σ_K the list of nodes of K, and V_h, the space of the *shape functions*, here it will be the Lagrange polynomial functions (or interpolants). To properly define the geometry and these functions, a reference space (that can also be seen as a parameter space) is defined where all coordinates are between 0 and 1. In this space, the reference element is denoted $\widehat{K}$ and has a fixed (and sometimes uniform) distribution of nodes. The element K, also called physical element, is thus the image of $\widehat{K}$ via a mapping, denoted F_K (see Figure 1). More specifically, for each point M of K, there is a point $\widehat{M}$ of $\widehat{K}$ such that $M = F_K(\widehat{M})$.

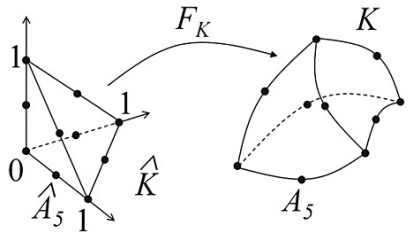

Fig. 1 Mapping F_K from $\widehat{K}$ to K.

Finally, the position of a point M inside K is defined by

$$M = \sum_{i=1}^{n} \phi_i(\widehat{M}) A_i,$$

where n is the number of nodes, $A_i = F_K(\widehat{A}_i)$ with $\widehat{A}_i$ the nodes of the reference element which map to A_i the nodes of the physical element, and ϕ_i are the Lagrange polynomial functions defined such that:

$$\phi_i(\widehat{A}_j) = \delta_{ij} \quad \text{and} \quad \sum_{i=1}^{n} \phi_i = 1.$$

It is important to note that in order to define a complete finite element of degree k on a simplex of dimension d (edge for $d = 1$, triangle for $d = 2$, tetrahedron for $d = 3$), the number of (distinct) nodes needs to be equal to $\frac{\Pi_{j=1}^{d}(k+j)}{d!}$. Also, on a simplex, the reference coordinates $(\widehat{x}, \widehat{y}, \widehat{z})$ can be used to define the simplex barycentric coordinates (u, v, w, t). For instance, for a triangle we have: $u = 1 - \widehat{x} - \widehat{y}$, $v = \widehat{x}$ and $w = \widehat{y}$, and for a tetrahedron $u = 1 - \widehat{x} - \widehat{y} - \widehat{z}$, $v = \widehat{x}$, $w = \widehat{y}$ and $t = \widehat{z}$.
A point M of the simplex K can also be expressed in Bézier form using the Bernstein polynomials B_{ijlm}^{d} as:

$$M = \sum_{i+j+l+m=k} B_{ijlm}^{d}(u, v, w, t) P_{ijlm},$$

with $B_{ijlm}^{d}(u, v, w, t) = \frac{d!}{i!j!l!m!} u^i v^j w^l t^m$. For a triangle, $m = 0$. The points $(P_{ijlm})_{i+j+l+m=k}$, also called $(C_i)_{1 \leq i \leq n}$ (see Figure 2) are the Bézier control points and are directly related to the nodes $(A_i)_{1 \leq i \leq n}$.
For instance, to compute C_5 (P_{1100}) in Figure 2, we use the formula (see [3] for details):

$$C_5 = \frac{4A_5 - A_1 - A_2}{2}.$$

In general and in the following sections, $(A_i)_{1 \leq i \leq d+1}$ are called vertices, $(A_i)_{d+2 \leq i \leq n}$ are called nodes, and $(C_i)_{1 \leq i \leq n}$ are called control points.

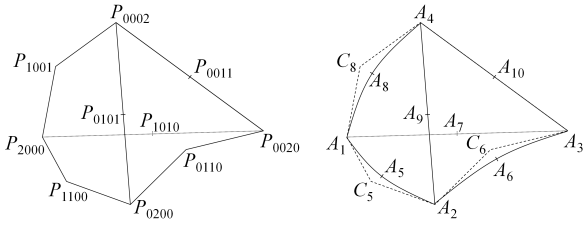

Fig. 2 Correspondence between the control points and the nodes for a P^2 tetrahedron.

The validity of an element means that the associated mapping F_K is a diffeomorphism. It can be ensured if the minimum of the determinant $\mathcal{J}_K$ of the Jacobian matrix of the mapping F_K is strictly positive everywhere inside the element [7]. In the case of a simplicial element, Jacobian $\mathcal{J}_K$ can

be written as polynomial of degree $d \times (k-1)$ in the barycentric coordinates of the simplex, where d is the dimension of the simplex and k the degree of the simplex. When the element is of degree 1 (e.g. straight-sided), it simply means that the oriented volume/area is strictly positive. The Jacobian can also be expressed in the Bernstein polynomial basis:

$$\mathcal{J}_K(u,v,w,t) = \sum_{i+j+l+m=d(k-1)} B_{ijlm}^{d(k-1)}(u,v,w,t) N_{ijlm}^K,$$

where N_{ijlm}^K are the control coefficient of the Jacobian. These coefficients can be explicitly found and have a geometrical meaning (see [3] for more details). In the case of a P^2-triangle (see Figure 3), a corner and an edge control coefficients [3] are for example:

$$N_{200}^K = 4 \det(\overrightarrow{P_{200}P_{110}}, \overrightarrow{P_{200}P_{101}}), \tag{1}$$

$$N_{110}^K = 2 \det(\overrightarrow{P_{200}P_{110}}, \overrightarrow{P_{110}P_{011}}) + 2 \det(\overrightarrow{P_{110}P_{020}}, \overrightarrow{P_{200}P_{101}}). \tag{2}$$

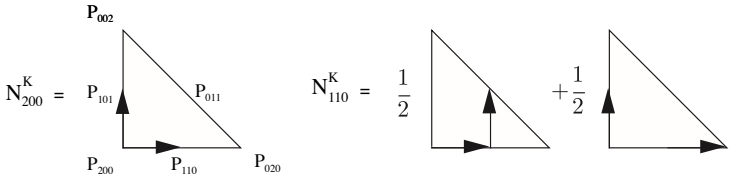

Fig. 3 Vectors involved in the determinant for the computation of a corner (left) and an edge (right) control coefficients of a P^2 triangle.

Consequently, a sufficient condition to prove that $\mathcal{J}_K$ is strictly positive everywhere is to ensure that all N_{ijlm}^K are strictly positive, but this is a too strong condition. On the contrary, if a N_{ijlm}^K is negative in a corner, it means that $\mathcal{J}_K$ is negative somewhere inside the element as N_{ijlm}^K is the exact value of the Jacobian in this corner [7]. However, if a control coefficient lying on an edge or on a face or in a volume is negative, it does not mean that $\mathcal{J}_K$ is negative somewhere inside the element. We cannot conclude on the positiveness of the Jacobian without any further analysis. In this case, a few iterations of a De Casteljau's algorithm [1, 7] (or a subdivision of the element [11]) are required to have more accurate bound of the Jacobian. The idea of this refinement on an edge/face/volume is to create new control coefficients from the initial ones. These control coefficients can define several curves whose union is the Jacobian curve on the edge/face/volume. This way, a more accurate bound can be found.

For instance, let us consider an edge on a P^2 tetrahedron (see Figure 4 on the left) with a negative edge control coefficient on it. The De Casteljau's algorithms can be decomposed in three steps (see Figure 4, middle, K superscripts are omitted for clarity):

$$N^1_{3000} = \frac{N_{3000} + N_{2100}}{2} \quad N^1_{2100} = \frac{N_{2100} + N_{1200}}{2} \quad N^1_{1200} = \frac{N_{1200} + N_{0300}}{2},$$
$$N^2_{3000} = \frac{N^1_{3000} + N^1_{2100}}{2} \quad N^2_{2100} = \frac{N^2_{2100} + N^2_{1200}}{2},$$
$$N^3_{3000} = \frac{N^2_{3000} + N^2_{2100}}{2}.$$

By construction, $N^3_{3000} = \mathcal{J}^K(\frac{1}{2}, \frac{1}{2}, 0, 0)$. So, if $N^3_{3000} \leq 0$ then the element is invalid. Otherwise, $N_{3000}, N^1_{3000}, N^2_{3000}, N^3_{3000}$ (resp. $N^3_{3000}, N^2_{2100}, N^1_{1200}, N_{0300}$) defines a P^3 curve representing the Jacobian between $\mathcal{J}^K(1,0,0,0)$ and $\mathcal{J}^K(\frac{1}{2}, \frac{1}{2}, 0, 0)$ (resp. $\mathcal{J}^K(\frac{1}{2}, \frac{1}{2}, 0, 0)$ and $\mathcal{J}^K(0,1,0,0)$) (see Figure 4, right). Consequently, the initial analysis can be done on these two sub-curves. If central control coefficients are positive, the Jacobian on the sub-element is valid. Otherwise, De Casteljau's algorithm is reapplied to this subcurve to conclude. If one subcurve gives an invalid Jacobian then the whole curve is invalid.
This way, a recursive method to find the sign of the minimum of the Jacobian on the edge is established.

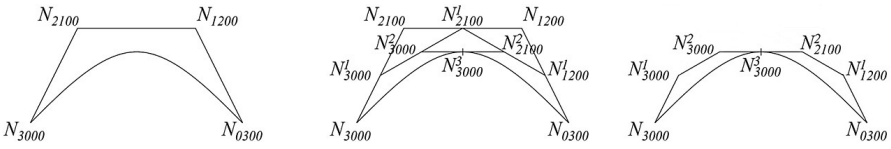

Fig. 4 De Castljau's refinement on two control coefficients of the Jacobian curve on an edge of a P^2 tetrahedron. Left, the initial curve with its control coefficients. Middle, construction of the new control coefficients. Right, the initial curve divided in two curves with their control coefficients.

Once the validity of the element is known, it is interesting to consider a quality criterion for the shape of the element. The chosen one is the one proposed in [8]:

$$Q = \underbrace{\alpha \frac{hS_k}{V_k}}_{①} \underbrace{\frac{\max(V_1, V_k)}{\min(V_1, V_k)}}_{②} \underbrace{\left(\frac{N^K_{max}}{N^K_{min}}\right)^{1/d}}_{③},$$

with:

- d the dimension, S_k the exterior surface of the polyhedron (in 2D, the half perimeter of the polygon) defined by nodes and vertices $(A_i)_{1 \leq i \leq n}$
- V_k the volume of the polyhedron (resp. surface of the polygon) defined previously
- h the element's largest edge P^k-length (e.g the length of the union of straight-sided lines defined by the nodes)

- V_1 the volume/surface of the equivalent P^1 element, e.g the element defined by the vertices
- N^K_{min} (resp. N^K_{max}) the smallest (resp. largest) control coefficient of the Jacobian of the element
- α is a normalization factor, dependent of the dimension such that $Q = 1$ for a regular simplex, $\alpha = \frac{\sqrt{3}}{6}$ in 2D and $\alpha = \frac{\sqrt{3}}{36}$ in 3D.

This quality function is actually a product of 3 terms. (1) is only a generalization of the P^1 quality function and measures the gap to the regular element. (2) measures the distance between the volume of the curved element and the volume of the straight element and ensures the function to be greater than 1 [8]. And, finally (3) gives a measure of the distortion of the element, it can detect if the element is invalid or almost invalid by taking an infinite value. Note that if the element is straight, the standard P^1 quality function [6] is recovered: $Q = Q_{P^1} = \alpha \frac{hS_1}{V_1} = \alpha \frac{h}{\rho}$ where ρ is the inradius of the straight element. Also, this quality function can easily be extended to anisotropic meshes. Based on these definitions, this element-wise quality measure is between 1 and the infinity. The closer the element quality is to 1, the better the quality is.

3 High-order mesh optimization

In the same way as we want to have an optimal P^1 mesh in terms of quality, we want to have an optimal P^k mesh. Several optimization techniques exist to correct an invalid P^k mesh [12, 16] and to optimize the geometrical accuracy [17].
The idea here is to extend two classic mesh quality-based optimization operators [1] to P^2 meshes to increase its quality.

3.1 P^2 swap operator

The swap operator (see Figure 5) locally changes the connectivity of the mesh in order to improve its quality.
In 2D, it consists in flipping an edge shared by two triangles to form two new triangles with the same four vertices (see Figure 5 left).
In 3D, two types of swap exist: face and edge swapping. The face swapping is the extension of the 2D edge swapping, it consists in replacing the common face of two neighboring tetrahedra by the edge linking the opposite vertices to the face of each tetrahedron, also called $2 \rightarrow 3$. The edge swapping is a bit different. First, the shell of the edge to delete (e.g. the set of elements containing this edge) is constructed. From a shell of size n, a non-planar pseudo-polygon formed by n vertices is obtained. The swap consists in deleting the edge, generating a triangulation of the polygon and creating two tetrahedra for each triangle of the triangulation thanks to the two extremities of the former edge. These swaps are designated as $n \rightarrow m$ with $n \geq 3$, where n is the initial number of tetrahedra and $m = 2(n-2)$ is the final number of tetrahedra.

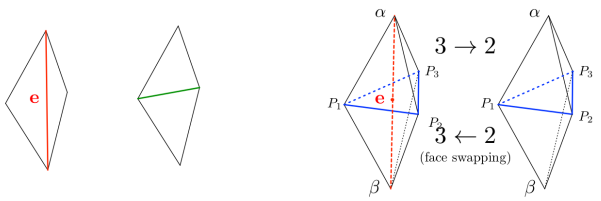

Fig. 5 *Left*, the swap operation in 2D. *Right*, edge swap $3 \to 2$ and face swap $2 \to 3$. For all these pictures, shells are in *black*, old edges are in *red*, new edges are in *green*.

For each possible swapped configuration, if the worst quality of all the elements the shell is improved, the configuration is kept and will be in the new mesh unless another swapped configuration of the shell provides a better quality improvement. When it comes to connectivity-change moving-mesh algorithms in 3D, a small local degradation of the shell's worst quality has been observed as an efficient way to improve the global quality of the mesh. To generalize it to P^2 meshes, the inner nodes of the edges of the shell have to be taken into account. For instance, for the P^2 case in 2D, there is one node on the swapped edge and if we want the swap to be performed, we need first to find an optimal position for the node in the swapped configuration and then to check if this configuration improves the quality function (see Figure 6). The key feature is therefore to find a functional whose optimum will give the optimal position for the node in the swapped configuration in term of quality.

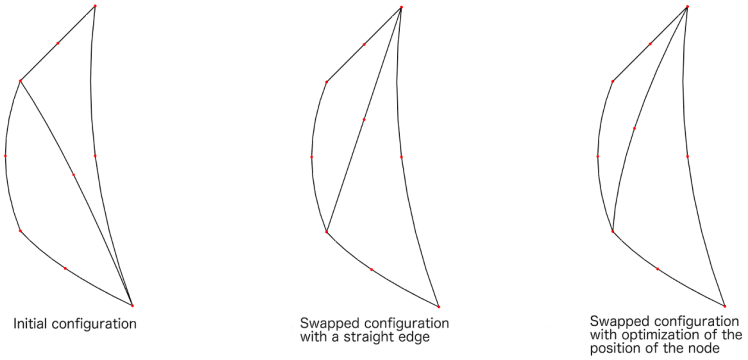

Fig. 6 Three steps of P^2 swap in 2D.

In this context, the idea is to find a simple and smooth functional that will be easy to optimize. The quality function is not a good candidate as it is not smooth. We propose to consider the following functional (inspired from the work of [16]):

$$f(\mathbf{X}) = \sum_{K \in \mathcal{S}(e)} \sum_{i+j+k+l=d} \omega_{ijkl} \left(\frac{N^K_{ijkl}(\mathbf{X})}{d! V_1} - 1 \right)^2,$$

where d is the dimension, $\mathbf{X} = [\mathbf{x_1}, ..., \mathbf{x_n}]$ is a *set* of n coordinates to be optimized, $\mathbf{X} \rightarrow N_{ijk}^K(\mathbf{X})$ is a function that depends, in the worst case on two variables of $\mathbf{X}$, $\mathcal{S}(e)$ is the shell of the initial edge e (e.g. the set of elements K containing e), ω_{ijkl} is a weight function that measures the importance of each control coefficient, and V_1 is the volume of the straight-sided element deduced from K. Note that $\mathbf{X}$ can either be empty (in which case no optimization is required), or contain more than one node's coordinates as it represents the coordinate of the created edges of the shell.
In 2D, $\mathbf{X}$ is always a singleton and is simply noted $\mathbf{x}$, weights ω_{ijk} $(l = 0)$ are equal to 2 for the corner coefficients and equal to 1 everywhere. In this case, f has the following properties : it is a positive define quadratic form as $\mathbf{x} \rightarrow N_{ijk}^K(\mathbf{x})$ is linear in $\mathbf{x}$, which means that the functional as a unique minimum. Also, on every regular swap configuration, the minimum of f in $\mathbf{x}$ is the same as the minimum of the worst quality of the swapped shell in $\mathbf{x}$. Using the result of the optimization problem in the swap configuration gives therefore a very good approximation of the best configuration that can be obtained. Since the best swap configuration is found, we are able to conclude if this swap will increase the quality or not.
In 3D, weights ω_{ijkl} are equal to 4 for a corner control coefficient, equal to 2 for an edge control coefficient and equal to 1 otherwise. In this work, considered swaps are $2 \rightarrow 3$, $3 \rightarrow 2$ and $4 \rightarrow 4$ which means that the functional of the problem is in the worst case quadratic. This is a consequence of the fact that control coefficients have a linear dependence with repect to a given control coefficient. Also, the problem appears numerically to be definite positive on a regular configuration. Note that these swaps represent $\sim 95\%$ of the swaps performed during a P^1 mesh optimization. For the other swaps ($5 \rightarrow 6$, $6 \rightarrow 8$), the problem begins to be highly costly in term of CPU. The resolution of these optimization problems is performed thanks to a L-BFGS algorithm [13]. Note that even if the quality function is not used for the optimization problem, it is mandatory to keep using it so that it degenerates into classical swap operator when the elements are straight.

3.2 P^2 mesh smoothing

Mesh smoothing is a technique that consists in relocating some points inside the mesh to improve the quality of the elements. In P^1, the idea is to relocate each vertex P_i inside its ball of elements (see Figure 7). For each element K_j in the ball of P_i, the opposite face to P_i denoted by F_j gives an optimal position P_j^{opt}. Then, the vertex is relocated considering a weighted average of the proposed positions. If the proposed new location of the vertex does not improve the ball configuration in term of quality, then a relaxation is performed to check if an improved configuration exists between the original location and the new one. The optimal configuration is computed as follows:

$$P_j^{opt} = G_j + \sqrt{\frac{2}{3}} h_j \frac{\mathbf{n}_j}{||\mathbf{n}_j||},$$

where G_j is the gravity center of F_j, h_j is the average length of the edges of F_j, and $\mathbf{n}_j$ is the outward normal to F_j. The proposed position is then computed with:

$$P_i^{opt} = \frac{\sum_{K_j \in Ball(P_i)} \max(Q(K_j), Q_{\max}) P_j^{opt}}{\sum_{K_j \in Ball(P_i)} \max(Q(K_j), Q_{\max})},$$

where $Q_{\max}$ is a parameter to be defined. Here $Q_{\max} = 10$. Note that if every computed configuration decreases the quality, the smoothing is not performed.

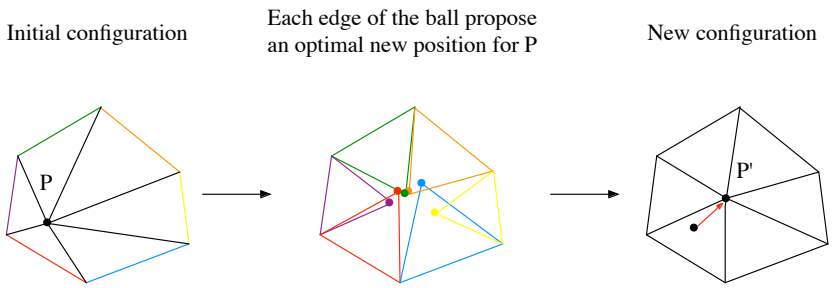

Fig. 7 Laplacian smoothing in two dimensions. Each element of the ball of considered vertex P_i suggests an optimal position for P_i. The resulting new optimal position for P_i is computed as a weighted average of all these proposed locations.

To extend it to P^2 meshes, the edges' node needs to be taken into account. The idea here, is to perform two independent smoothing operations:

- A vertex smoothing

The vertex smoothing is simply a generalization of the P^1 smoothing. The optimal position of the vertex is computed in the same way as in P^1, and the vertex is located exactly in the same way as before. In order to be consistent with the P^1 vertex smoothing and to keep in the ball straight edges that are initially straight, the displacement of all the inner nodes of the ball cavity is set to half of the value of the displacement of the central vertex (see Figure 8). In the exact same way as in P^1 if the final configuration does not improve the quality, relaxation is performed the original location and the new one.

- A node smoothing.

The optimization of the node position follows the same algorithm as in the P^2 swap operator to find its optimal position. For this purpose, functional f can be re-used to find the optimal node position (see Figure 9). In this case, there is always only one node coordinates to optimize and consequently the optimization problem is quadratic. In the exact same way as in P^1, if the final configuration does not improve the quality, relaxation is performed the original location and the new one.

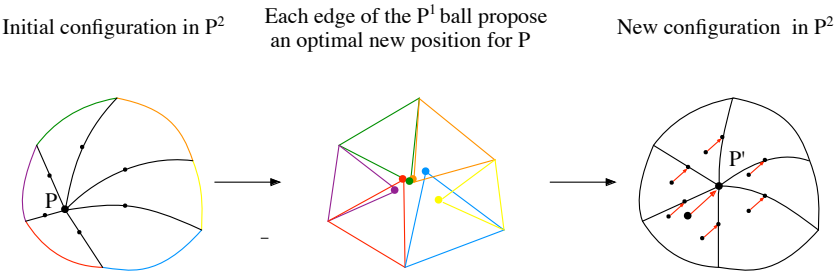

Fig. 8 P^2 laplacian smoothing in two dimensions. Each element of the P^1 ball of considered vertex P suggests an optimal position for P. The resulting new optimal position for P is computed as a weighted average of all these proposed locations. The new position of the nodes of the internal edges of the P^2 ball is then deduced by proportionality.

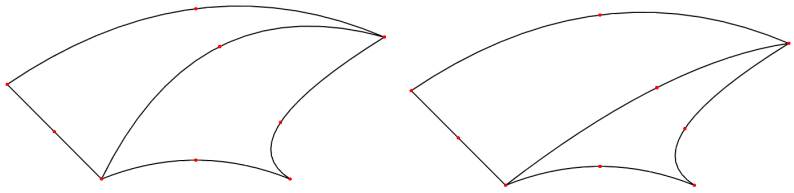

Fig. 9 P^2 node smoothing in two dimensions. The optimal position of the node of the central edge is computed solving an optimization problem. Left, the initial configuration, right, the optimized configuration.

4 Applications

4.1 High-order mesh generation by curving an initial P^1 mesh

Most of the techniques to generate an high-order mesh is to start from a P^1 mesh and then to curve it, in a way or another, in order to obtain a P^k mesh [5, 12, 14, 18]. The main reason to use a post-treatment is that all existing P^1 mesh generation algorithms can be reused. It would be harder to implement a directly high-order mesh generator. To curve meshes, used models are numerous : PDE or variational models [5, 9, 14], smoothing and/or optimization procedures [12, 15, 16], ... Our choice here is to use the linear elasticity equation as a model for the motion of the vertices to generate a P^k mesh from a P^1 mesh.
For this purpose, let us consider the linear elasticity equation with Dirichlet boundary conditions:

$$\nabla \cdot (\sigma(\mathcal{E})) = 0\,, \quad \text{with} \quad \mathcal{E} = \frac{\nabla\xi + {}^T\nabla\xi}{2}\,, \tag{3}$$

where σ, and $\mathcal{E}$ are respectively the Cauchy stress and strain tensors, ξ is the Lagrangian displacement. The Cauchy stress tensor follows the Hooke's law for isotropic homogeneous medium.
Here, Dirichlet boundary conditions represent the gap between the P^k-nodes of the initial straight boundary elements and their position on the *real* boundary. For mesh boundary vertices, the gap is equal to 0.
To compute the gap at the nodes, a continuous representation of the surface mesh is required. It can be either provided by CAD/analytical model or deduced from initial P^1 mesh via a cubic reconstruction technique [20].
Once Dirichlet boundary conditions are set, the high-order finite element linear elasticity code is called. The use of an high-order FE resolution rather than on a subdivided P^1 mesh aims the degrees of freedom to be intrinsically represented. This gives more consistency to the obtained motion.
The elasticity problem using the high-order FEM provides the new position of the internal vertices and nodes. It is then used to generate the high-order mesh by moving the vertices and nodes of the initial straight mesh with the associated values in the elasticity solution vector. If some elements remain invalid after optimization, it is always due to non suitable boundary displacements. In this case, the FEM solution is proportionally reduced in the vicinity (boundary included) of the invalid element until global validity is obtained. The process is summarized by Algorithm 1.

Algorithm 1 Mesh curving algorithm

1. Generate a P^1 mesh.
2. Perform P^1 mesh optimization pre-processing: generalized swapping and vertex smoothing.
3. Perform cubic reconstruction of the boundary or use its analytical representation to set Dirichlet boundary conditions for the linear elasticity equation.
4. Solve linear elasticity equation on the P^1 mesh with the FEM at the order of the wanted mesh.
5. Generate the P^k mesh by moving the P^1 mesh with the solution of the linear elasticity.
6. Perform P^2 mesh optimization post-processing: generalized swapping and node/vertex smoothing.
7. Check validity of P^k elements and locally relax the FEM solution if necessary or desired until it is valid.

The major fact with this method is that the deformed mesh is only made of isotropic or almost isotropic elements. In this context, the use of the elasticity problem is efficient and always provides a valid mesh. Optimization in the pre-processing makes elements more isotropic and therefore helps curvature process to be more robust whereas optimization in the post-processing improves the quality of the mesh and untangle invalid elements if any. Some results in P^2 are presented below:

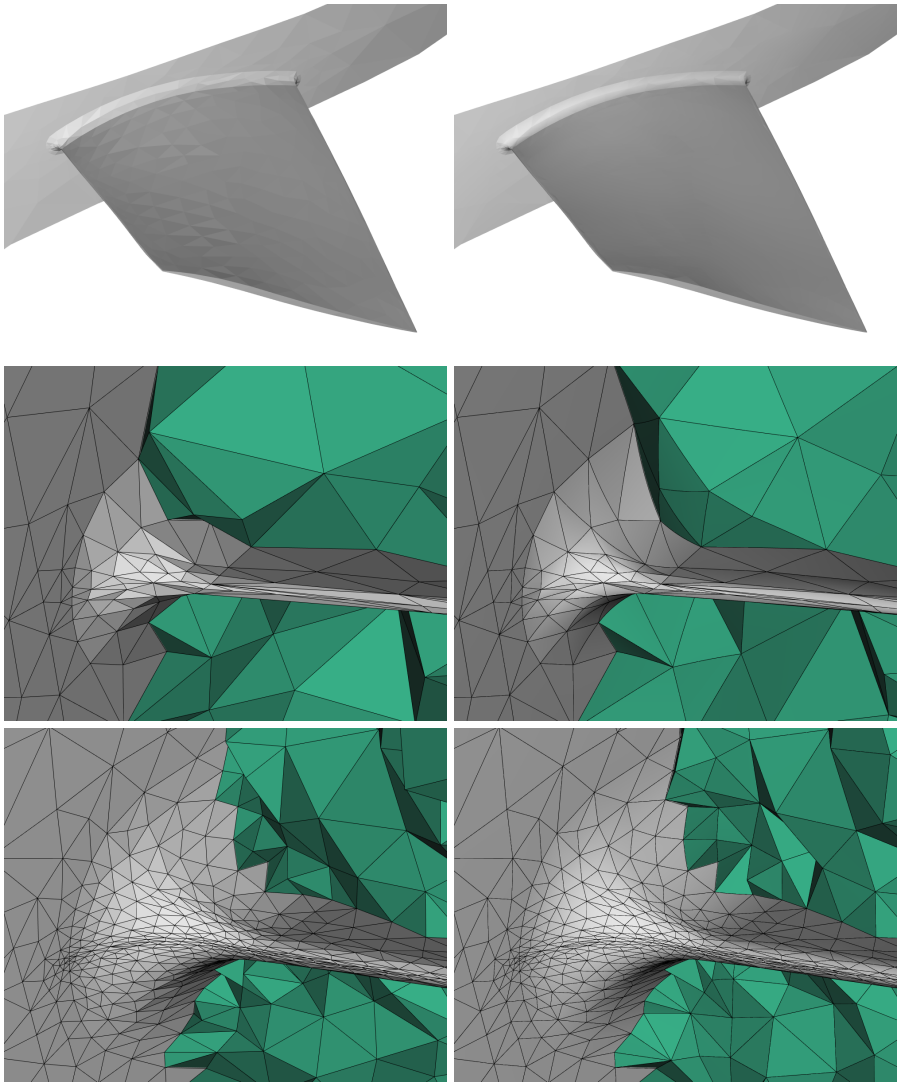

Fig. 10 NASA RO37 rotor turbomachine meshes. From top to bottom, surface, coarse and fine meshes. From right to left, P^1 and P^2 meshes. P^2 meshes are generated with Algorithm 1 using a cubic reconstruction.

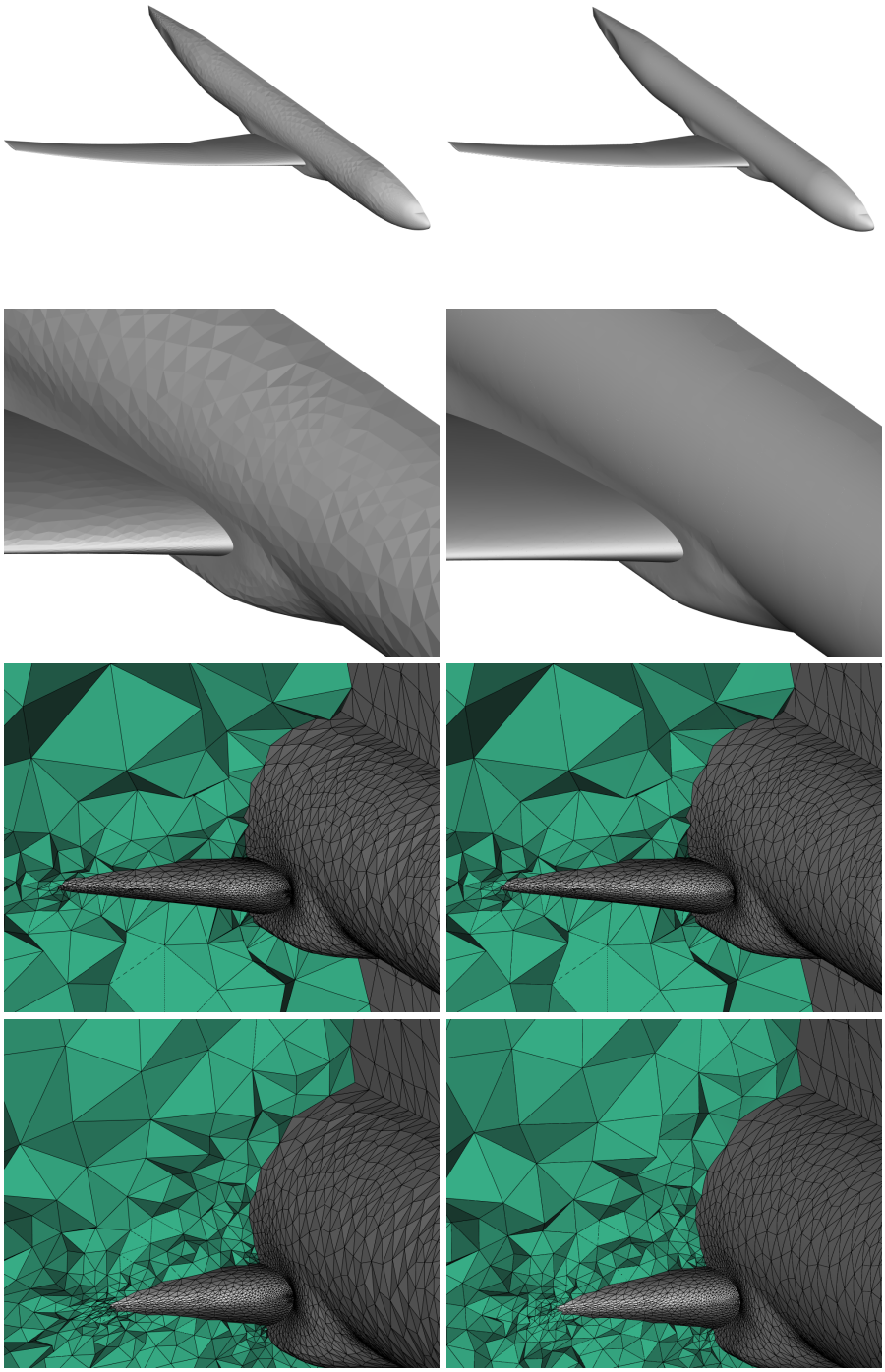

Fig. 11 NASA Common Research Model aircraft meshes. From top to bottom, surface, coarse and fine meshes. From right to left, P^1 and P^2 meshes. P^2 meshes are generated with Algorithm 1 using a cubic reconstruction.

4.1.1 NASA RO37 rotor

This example is a NASA RO37 rotor used for turbomachinery applications. Two different meshes are considered (see Figure 10).

- A coarse mesh (see Figure 10, middle) with an initial number of 2 434 vertices, 13 326 nodes and 10 970 tetrahedra. The initial mesh average quality is 6.16with a worst quality of 1 682 due to the presence of sharp anisotropic trailing and leading edges on the input wing surface mesh. Curving the mesh without optimization provides an average quality of 5.3 with a worst quality of 1 729. Optimization in post and pre processing increase average quality to 2.5 and worst quality to 1 346. Note that the number of nodes and tetrahedra is changed to respectively 11 598 and 10 862.
- A fine mesh (see Figure 10, bottom) with an initial number of 22 145 vertices, 137 393 nodes and 106 562 tetrahedra. The initial mesh average quality is 2.14 with a worst quality of 111. Note that is better than with the coarse mesh as the mesh is finer and therefore deals better with trailing and leading edges. Curving the mesh without optimization provides an average quality of 2.15 with a worst quality of 366 . Optimization in post and pre processing increase average quality to 1.7 and worst quality to 27.5. Note that the number of nodes and tetrahedra is changed to respectively 136 924 and 106 093.

In both cases, we clearly observe the benefits of the optimization that improves the average and the worst quality of the final mesh. Note that the worst quality of the final P^2 might be greater than the one of the initial P^1 mesh. This a consequence of the curving process that decreases the quality of straight elements by curving them. We can also observe that the curvature is not propagated a lot in the volume as it is not visible after 2 or 3 layers of elements (see Figure 10, middle and bottom right). This is an illustration of St Venant's principle which states that the elasticity solution can be divided into transmissive effects and local disturbances.

4.1.2 NASA Common research model aircraft

This example is the NASA Common Research Model, an aircraft model that is massively used in both experimental and numerical simulations in fluid dynamics (see Figure 11). Again, two meshes are considered:

- A coarse mesh (see Figure 11, line 3) with an initial number of 32 479 vertices, 173 468 nodes and 118 012 tetrahedra. The initial mesh average quality is 2.49 with a worst quality of 271 due to the presence of sharp anisotropic trailing and leading edges on the input wing surface mesh. Curving the mesh without optimization provides an invalid configuration with 10 invalid elements. Optimization in post and pre processing increase average quality to 2.13 and worst quality to 271. Note that the number of nodes and tetrahedra is changed to respectively 173 744 and 118 288.
- A fine mesh (see Figure 11, line 4) with an initial number of 101 422 vertices, 660 071 nodes and 535 672 tetrahedra. The initial mesh average

quality is 2.28 with a worst quality of 1 314. Curving the mesh without optimization provides an invalid configuration with 4 invalid elements. Optimization in post and pre processing increase average quality to 1.4 and worst quality to 1 777. Note that the number of nodes and tetrahedra is changed to respectively 659 545 and 535 146.

In both cases, optimization ensures a valid curved mesh at the end that would not have been obtained without it. In the same way as in the previous example, the curvature is not propagated a lot in the volume as it is not visible after 2 or 3 layers (see Figure 11, line 3 and 4 right).

4.2 A moving mesh technique for high-order elements

In this section, a connectivity-change moving-mesh method inspired from [1] for P^2 meshes is presented. In this case, the initial mesh is a P^2-mesh whose boundary has an initial displacement. Using a linear elasticity analogy, the resolution of the elasticity equation with high-order finite elements gives us a displacement for all the vertices and nodes in the volume. Then the mesh is moved to the new position. The motion of the vertices can be also enhanced by using a local stiffness factor technique [1]. This technique locally multiplies the tensor σ of linear elasticity equation by a factor proportional to $\mathcal{J}_K(\mathbf{x})^{-\chi}$. χ determines the degree by which smaller elements are rendered stiffer than larger ones. We use $\chi = 1$ [1]. Afterwards, connectivity changes are performed on the mesh to improve the *quality* of the elements. It is an efficient way to get rid of any shearing that occurs in the mesh. The high-order moving-mesh algorithm is summarized in Algorithm 2.

Algorithm 2 High-order moving mesh algorithm

1. Mesh deformation algorithm.
   a. Compute body displacement from body translation and rotation data.
   b. Solve linear elasticity equation with the FEM at the order of the mesh.
   c. Perform high-order mesh optimization.
   d. Check validity of the obtained displacement and restart it with a smaller body displacement if necessary/desired until the obtained displacement is valid.
2. Move the mesh.

The studied case is a moving sphere of radius 0.6 inside a large control volume (see Fig 12, line 1 left). At each iteration, the sphere is displaced of 0.08 in x direction. The initial mesh (see Fig 12, line 2) has an average quality of 1.27 and a worst quality of 2.5. Three positions are considered (see Fig 12, line 1 right) with and without connectivity-change: the initial position, the position after 30 iterations (displacement of 4 radii) and the position after 50 iterations (displacement of 6.7 radii).

When no connectivity-change is done after each iteration, shearing appears in the mesh which constraints the displacement. The high-order linear elasticity resolution gives to the elements a curvature that fits to the displacement of

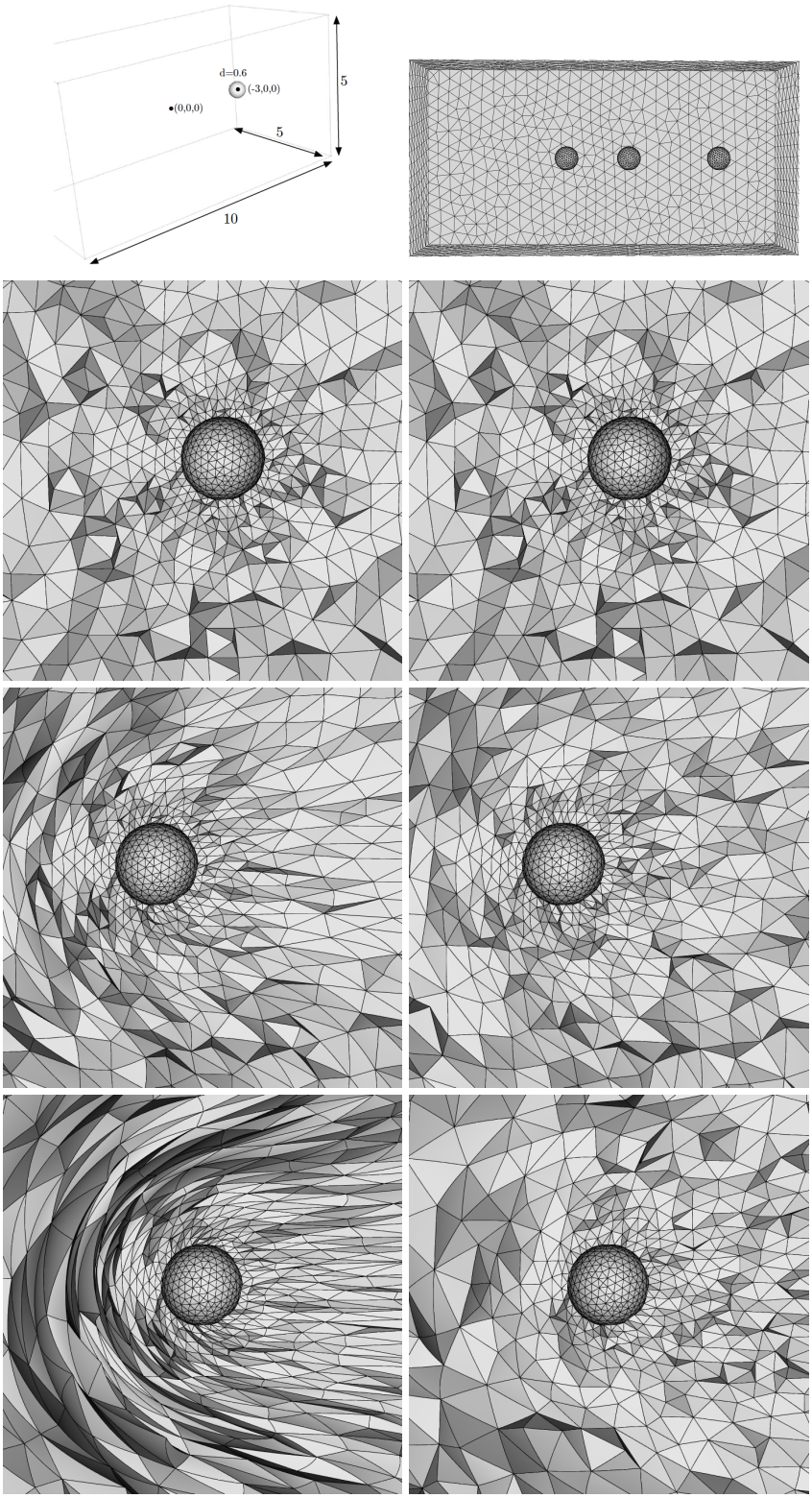

Fig. 12 Case of the moving sphere inside a P^2 mesh. First line, description of the sphere and description of the three considered positions of the moving sphere: the sphere is moving from right to left. And then, from top to bottom, zoom in the vicinity of the sphere for each position. Left, without mesh optimization operators. Right, with mesh optimization operators.

the sphere in order to move as far as possible the object when the mesh connectivity is fixed. Indeed, in front of the sphere, the deformed elements fit to the shape of the sphere, whereas in the wake, the curvature of elements is made so that the shearing is reduced. This is a good point but the mesh quality decreases drastically (see Fig 12, line 3 and 4 left): after 30 iterations, average quality is 1.64 and worst quality is 5.7 and after 50 iterations, 84 elements are invalid. On the contrary, when mesh quality-based optimization operators are considered with the moving mesh algorithm at each iteration, the average and the worst quality do not change a lot (see Fig 12, line 3 and 4 right): after 30 iterations, average quality is 1.43 and worst quality is 3.1 and after 50 iterations, average quality is 1.48 and worst quality is 3.5.

5 Conclusion and perspectives

P^2 mesh quality-based optimization operators have been presented. These operators ensure a valid P^2 mesh generation starting from a P^1 mesh and enable to deal with connectivity-change P^2 moving-mesh methods. Note that all these developments are suitable for isotropic meshes but do not work that well on anisotropic meshes and do not work as is with boundary layer meshes. The moving-mesh method gives similar results in term of quality as in P^1 which is promising for future research. The immediate next step will be to generalize it to curved trajectories using [1]. Isotropic degree two meshes were only the first step, further developments will be to generalize it to any higher-order meshes and to deal with anisotropy using metric fields. When it comes to boundary layer meshes, the goal is to extend the closed-advancing boundary layer mesh generation method of [2] to high-order meshes in order to generate directly curved boundary layer meshes. To this end, it is required to:

- Start form an initial high-order mesh that is obtained using the method of Section 4.1
- Consider the connectivity-change moving mesh method for high-order mesh presented in Section 4.2 with curved trajectories to deform the initial high-order mesh when the boundary layer mesh is inflated inside the domain
- Generate directly high-order elements in the boundary layer when it is inflated using the advancing layer approach presented in [2].

The future work to do is the last item. The advancing layer method will be modified to take into account the high-order boundary layer elements. The new position of the nodes in the boundary layer will be given using the same process as the one for proposing the new position for the vertices. High-order quality functions will be used to check the quality of the boundary layer elements when they are generated.
Note that more accurate normals will be obtained as they will be computed on the high-order mesh instead of a P^1-straight mesh. This is an important point as the quality of the boundary layer is highly dependent on the accuracy of

the normal computations [2]. Also, curved boundary layer mesh generation will not need the creation of a new tool and should be more robust and efficient as it directly controls the high-order mesh quality.

6 Acknowledgment

This work was supported by a public grant as part of the *Investissement d'avenir* project, reference ANR-11-LABX-0056-LMH, LabEx LMH.
The authors also would like to thank the reviewers for their fruitful remarks.

References

[1] F. Alauzet. A changing-topology moving mesh technique for large displacements. *Engineering with Computers*, 30(2):175–200, 2014.

[2] F. Alauzet, A. Loseille, and D. Marcum. On a robust boundary layer mesh generation process. In *55th AIAA Aerospace Sciences Meeting*, AIAA Paper2017-0585, Grapevine, TX, USA, 2017.

[3] H. Borouchaki and P. L. George. *Meshing, Geometric Modeling and Numerical Simulation 1: Form Functions, Triangulations and Geometric Modeling*. John Wiley & Sons, 2017.

[4] P.G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland Publishing Company, 1978.

[5] M. Fortunato and P.-O. Persson. High-order Unstructured Curved Mesh Generation Using the Winslow Equations. *J. Comput. Phys.*, 307:1–14, February 2016.

[6] P. J. Frey and P. L. George. *Mesh generation: application to finite elements*. John Wiley & Sons, 2008.

[7] P. L. George and H. Borouchaki. Construction of tetrahedral meshes of degree two. *International Journal for Numerical Methods in Engineering*, 2012.

[8] P. L. George, H. Borouchaki, F. Alauzet, P. Laug, A. Loseille, and L. Maréchal. *Meshing, Geometric Modeling and Numerical Simulation 2: Metrics, Meshing and Mesh Adaptation*. John Wiley & Sons, 2018 (to appear).

[9] R. Hartmann and T. Leicht. Generation of unstructured curvilinear grids and high-order discontinuous Galerkin discretization applied to a 3D high-lift configuration. *International Journal for Numerical Methods in Fluids*, 82(6):316–333, 2016.

[10] J.S. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods: algorithms, analysis and applications*. Springer Publishing Company, Incorporated, 2008.

[11] A. Johnen, J.-F. Remacle, and C. Geuzaine. Geometrical validity of curvilinear finite elements. *Journal of Computational Physics*, 2013.

[12] S. L. Karman, J T. Erwin, R. S. Glasby, and D. Stefanski. High-order mesh curving using WCN mesh optimization. In *46th AIAA Fluid Dynamics Conference*, AIAA AVIATION Forum. American Institute of Aeronautics and Astronautics, 2016.

[13] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989.

[14] D. Moxey, D. Ekelschot, Ü. Keskin, S.J. Sherwin, and J. Peirò. High-order curvilinear meshing using a thermo-elastic analogy. *Computer-Aided Design*, 72:130 – 139, 2016.

[15] E. Ruiz-Gironès, X. Roca, and J. Sarrate. High-order mesh curving by distortion minimization with boundary nodes free to slide on a 3D CAD representation. *Computer-Aided Design*, 72:52 – 64, 2016. 23rd International Meshing Roundtable Special Issue: Advances in Mesh Generation.

[16] T. Toulorge, C. Geuzaine, J.-F. Remacle, and J. Lambrechts. Robust untangling of curvilinear meshes. *Journal of Computational Physics*, 254:8 – 26, 2013.

[17] T. Toulorge, J. Lambrechts, and J.-F. Remacle. Optimizing the geometrical accuracy of curvilinear meshes. *Journal of Computational Physics*, 310:361 – 380, 2016.

[18] M. Turner, J. Peirò, and D. Moxey. A Variational Framework for High-order Mesh Generation. *Procedia Engineering*, 163(Supplement C):340 – 352, 2016. 25th International Meshing Roundtable.

[19] J. Vanharen, G. Puigt, X. Vasseur, J.-F. Boussuge, and P. Sagaut. Revisiting the spectral analysis for high-order spectral discontinuous methods. *Journal of Computational Physics*, 337:379 – 402, 2017.

[20] A. Vlachos, P. Jörg, C. Boyd, and J. L. Mitchell. Curved PN Triangles. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, I3D '01, pages 159–166, New York, NY, USA, 2001. ACM.